

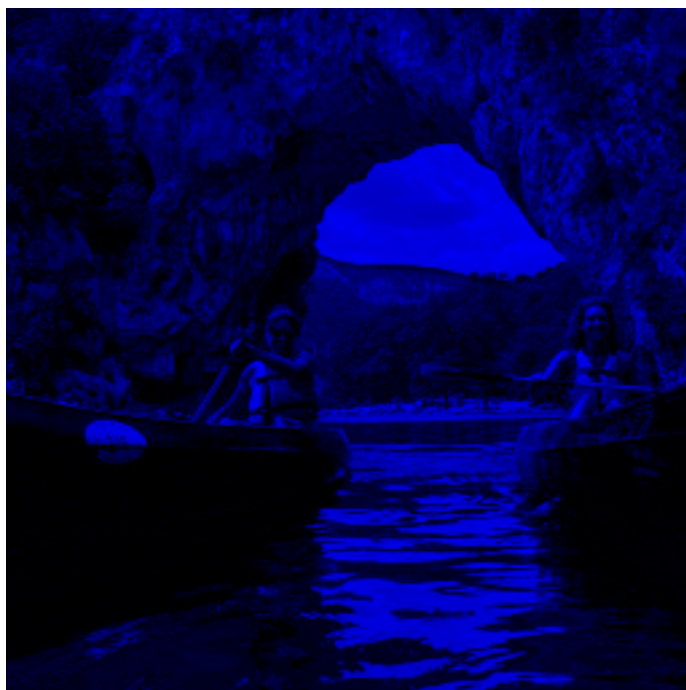
# Compte Rendu TP2

## Cossin Tristan

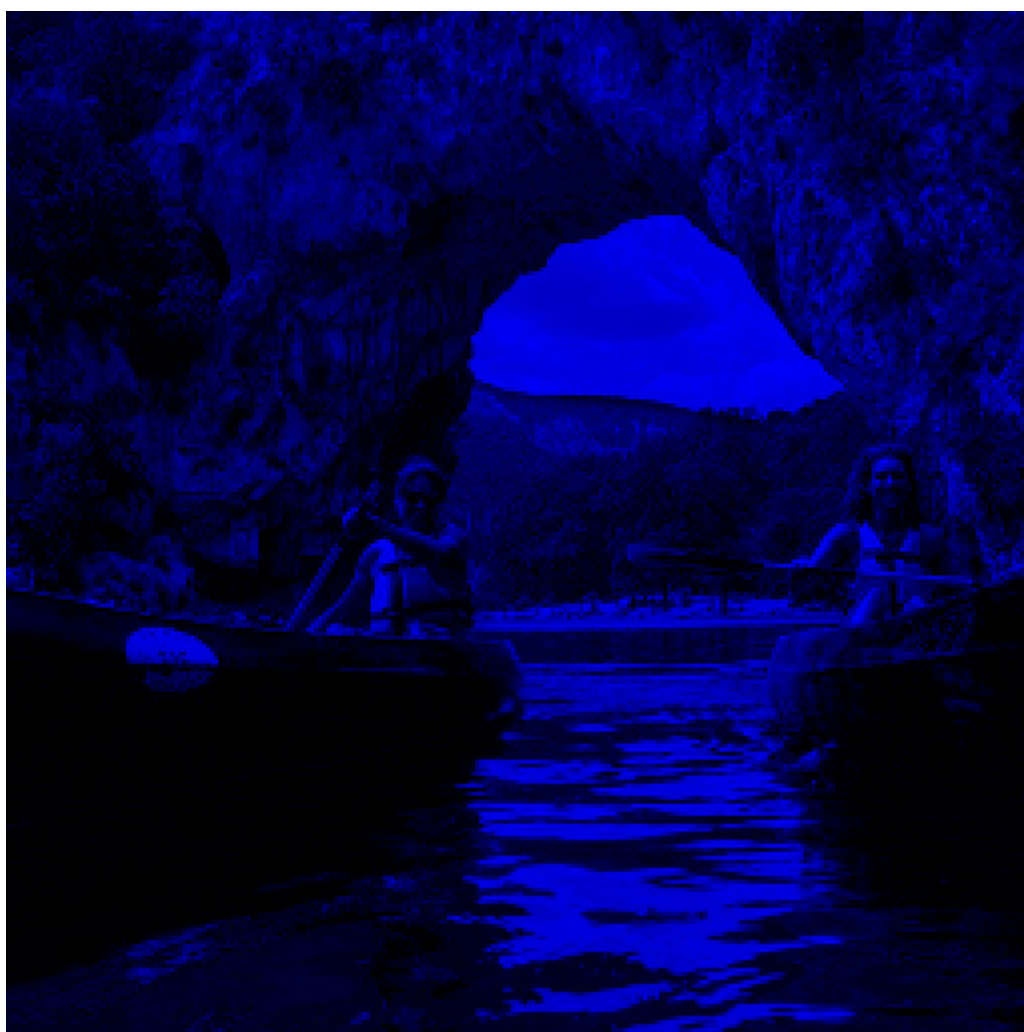
### Question 3 :



*Composante Rouge*



*Composante Bleu réduite*



*Composante Bleu ré-agrandi*



*Composante verte réduite*



*Composante verte ré-agrandi*

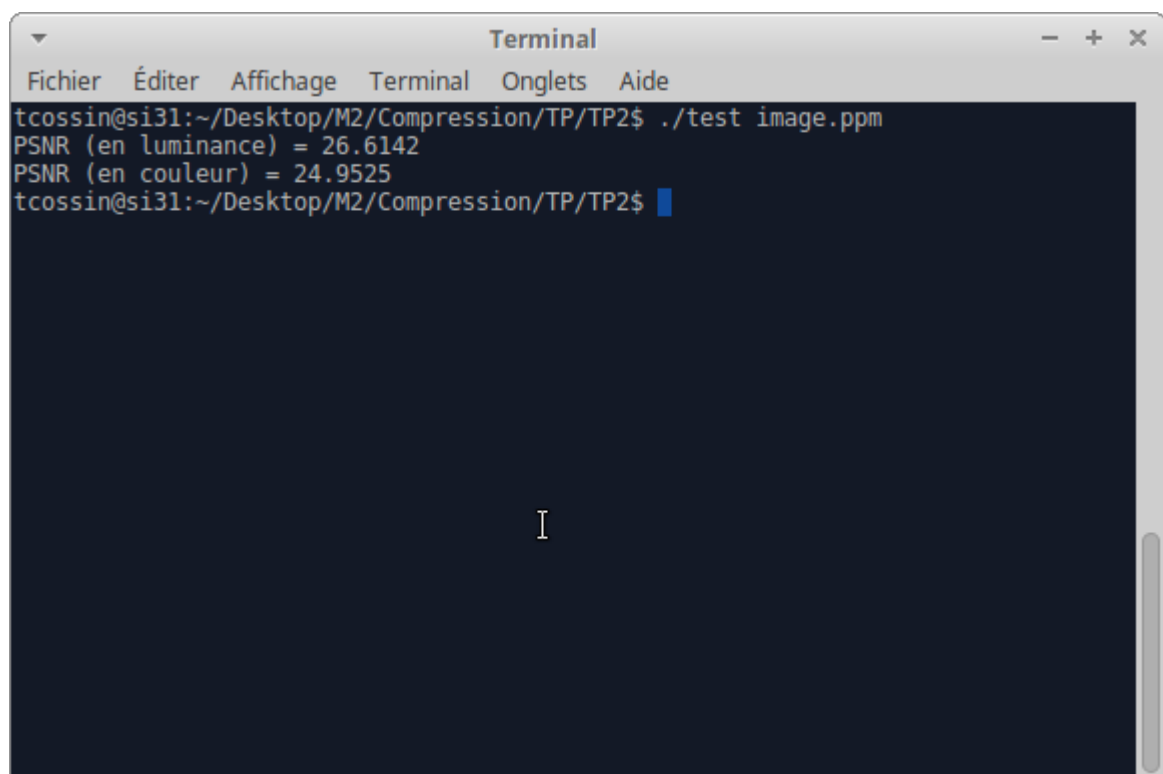




*Image originale*



*Image après ré-assemblage*



A terminal window titled "Terminal" with a menu bar containing "Fichier", "Éditer", "Affichage", "Terminal", "Onglets", and "Aide". The window has standard window controls (minimize, maximize, close) in the top right corner. The terminal content shows a user running a test script on a file named "image.ppm". The output displays two PSNR values: 26.6142 for luminance and 24.9525 for color. The prompt indicates the user is in the directory ~/Desktop/M2/Compression/TP/TP2.

```
tcossin@si31:~/Desktop/M2/Compression/TP/TP2$ ./test image.ppm
PSNR (en luminance) = 26.6142
PSNR (en couleur) = 24.9525
tcossin@si31:~/Desktop/M2/Compression/TP/TP2$
```

**Question 4 :**



*Composante Y*



*Composante CB réduite*



*Composante CB ré-agrandi*



*Composante CR réduite*



*Composante CR ré-agrandi*

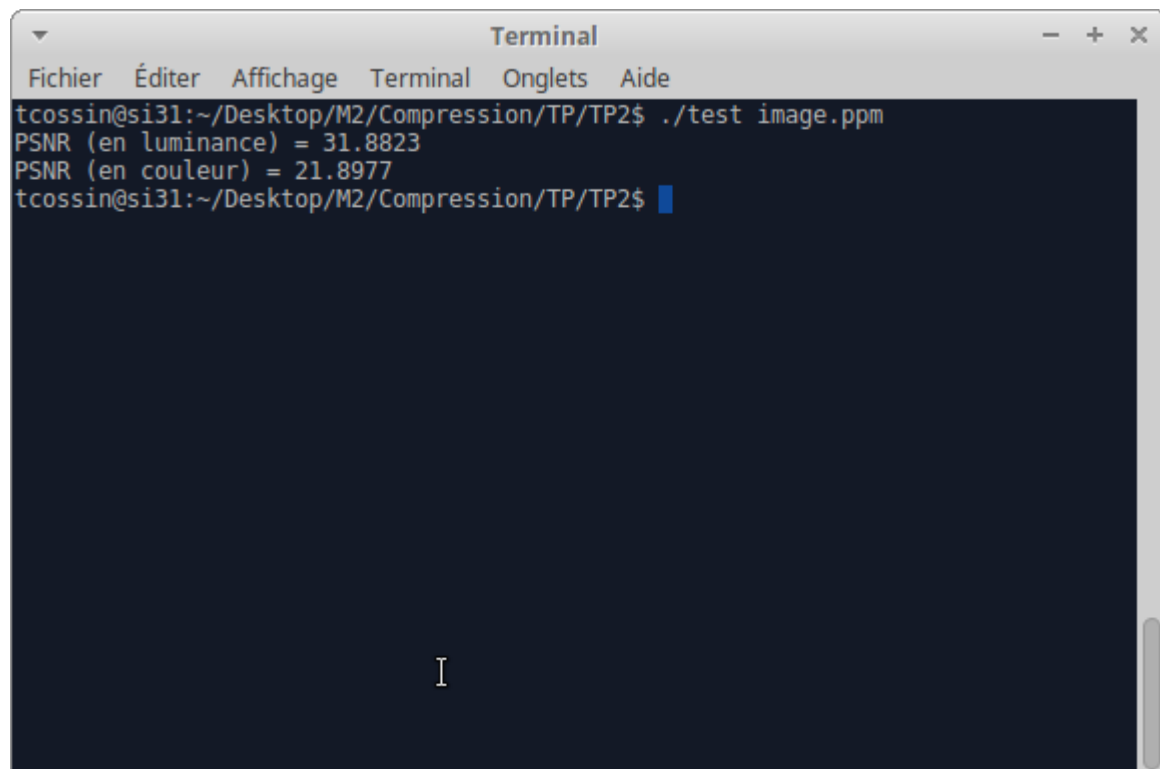




*Image originale*



*Image finale après ré-assemblage*

A screenshot of a Linux terminal window titled "Terminal". The window has a menu bar with "Fichier", "Éditer", "Affichage", "Terminal", "Onglets", and "Aide". The terminal content shows a user running a command to test image compression, resulting in two PSNR values: 31.8823 for luminance and 21.8977 for color. The prompt is tcossin@si31:~/Desktop/M2/Compression/TP/TP2\$.

```
tcossin@si31:~/Desktop/M2/Compression/TP/TP2$ ./test image.ppm
PSNR (en luminance) = 31.8823
PSNR (en couleur) = 21.8977
tcossin@si31:~/Desktop/M2/Compression/TP/TP2$
```

**Question 5 :**

Une technique d'approche pour avoir un taux de compression de 2 serait de diminuer les nuances de couleur du blanc et du noir. Ainsi en réduisant de 1 ou 2 bit au début et à la fin de chaque pixel, on peut obtenir un taux de compression satisfaisant.

## **Explications :**

Pour le ré-échantillonnage, j'ai dupliqué chaque pixel une fois sur la ligne pour réobtenir la taille originale et pour les lignes manquantes, j'ai dupliquer la ligne

```
int y_bis = 0;
```

```
//on agrandit de nouveau les canaux Vert et Bleu
```

```
for(int x = 0; x < Composante_Vert.getHeight(); x++)//hauteur
```

```
{
```

```
    for(int y = 0; y < Composante_Vert.getWidth(); y++)//largeur
```

```
    {
```

```
        if(y_bis == (imIn.getWidth()-1))
```

```
            y_bis = 0;
```

```
        //composante verte
```

```
        Composante_Vert_Out[x * 3][y_bis * 3] = 0; // R
```

```
        Composante_Vert_Out[x * 3][(y_bis * 3) + 1] = Composante_Vert[x * 3][(y * 3) + 1]; // G
```

```
        Composante_Vert_Out[x * 3][(y_bis * 3) + 2] = 0; // B
```

```
        Composante_Vert_Out[x * 3][(y_bis + 1) * 3] = 0; // R
```

```
        Composante_Vert_Out[x * 3][((y_bis + 1) * 3) + 1] = Composante_Vert[x * 3][(y * 3) +
```

```
1]; // G
```

```
        Composante_Vert_Out[x * 3][((y_bis + 1) * 3) + 2] = 0; // B
```

```
        Composante_Vert_Out[(x + 1) * 3][y_bis * 3] = 0; // R
```

```
        Composante_Vert_Out[(x + 1) * 3][(y_bis * 3) + 1] = Composante_Vert[x * 3][(y * 3) +
```

```
1]; // G
```

```
        Composante_Vert_Out[(x + 1) * 3][(y_bis * 3) + 2] = 0; // B
```

```
        Composante_Vert_Out[(x + 1) * 3][(y_bis + 1) * 3] = 0; // R
```

```
        Composante_Vert_Out[(x + 1) * 3][((y_bis + 1) * 3) + 1] = Composante_Vert[x * 3][(y * 3)
```

```
+ 1]; // G
```

```
        Composante_Vert_Out[(x + 1) * 3][((y_bis + 1) * 3) + 2] = 0; // B
```

```
        //composante bleu
```

```
        Composante_Bleu_Out[x * 3][y_bis * 3] = 0; // R
```

```
        Composante_Bleu_Out[x * 3][(y_bis * 3) + 1] = 0; // G
```

```
        Composante_Bleu_Out[x * 3][(y_bis * 3) + 2] = Composante_Bleu[x * 3][(y * 3) + 2]; // B
```

```
        Composante_Bleu_Out[x * 3][(y_bis + 1) * 3] = 0; // R
```

```
        Composante_Bleu_Out[x * 3][((y_bis + 1) * 3) + 1] = 0; // G
```

```
        Composante_Bleu_Out[x * 3][((y_bis + 1) * 3) + 2] = Composante_Bleu[x * 3][(y * 3) + 2];
```

```
// B
```

```
        Composante_Bleu_Out[(x + 1) * 3][y_bis * 3] = 0; // R
```

```
        Composante_Bleu_Out[(x + 1) * 3][(y_bis * 3) + 1] = 0; // G
```

```
        Composante_Bleu_Out[(x + 1) * 3][(y_bis * 3) + 2] = Composante_Bleu[x * 3][(y * 3) + 2];
```

```
// B
```

```
        Composante_Bleu_Out[(x + 1) * 3][(y_bis + 1) * 3] = 0; // R
```

```
        Composante_Bleu_Out[(x + 1) * 3][((y_bis + 1) * 3) + 1] = 0; // G
```

```
        Composante_Bleu_Out[(x + 1) * 3][((y_bis + 1) * 3) + 2] = Composante_Bleu[x * 3][(y * 3)
```

```
+ 2]; // B
```

```
y_bis = y_bis + 2; } }
```

### **Calcul du PSNR:**

```
//calcul du PSNR (luminance)
float som = 0;
for(int x = 0; x < imIn.getHeight(); x++)//hauteur
{
    for(int y = 0; y < imIn.getWidth(); y++)//largeur
    {
        float lumOut = (0.3 * Image_Finale[x*3][y*3+0]) + (0.6 * Image_Finale[x*3][y*3+1]) + (0.1 *
            Image_Finale[x*3][y*3+2]);
        float lumIn = (0.3 * imIn[x*3][y*3+0]) + (0.6 * imIn[x*3][y*3+1]) + (0.1 * imIn[x*3][y*3+2]);

        som += pow( lumIn - lumOut, 2);
    }
}
float EQM = som / (imIn.getHeight() * imIn.getWidth());
float PSNR = 10 * log10((255*255)/EQM);
```

```
//calcul du PSNR (luminance)
float Somme_Rouge = 0;
float Somme_Vert = 0;
float Somme_Bleu = 0;

for(int x = 0; x < imIn.getHeight(); x++)//hauteur
{
    for(int y = 0; y < imIn.getWidth(); y++)//largeur
    {
        Somme_Rouge += pow(imIn[x * 3][y * 3] - Image_Finale[x * 3][y * 3], 2);
        Somme_Vert += pow(imIn[x * 3][(y * 3) + 1] - Image_Finale[x * 3][(y * 3) + 1], 2);
        Somme_Bleu += pow(imIn[x * 3][(y * 3) + 2] - Image_Finale[x * 3][(y * 3) + 2], 2);
    }
}

float EQM_Rouge = Somme_Rouge / (imIn.getHeight() * imIn.getWidth());
float EQM_Vert = Somme_Vert / (imIn.getHeight() * imIn.getWidth());
float EQM_Bleu = Somme_Bleu / (imIn.getHeight() * imIn.getWidth());
EQM = EQM_Rouge + EQM_Vert + EQM_Bleu;
PSNR = 10 * log10((3 * pow(255, 2))/EQM);
```

### **Conversion YCrCb to RGB:**

```
r = y + 1.402(cr - 128)
g = y - 0.34414 (cb - 128) - 0.714414(cr - 128)
b = y + 1.772 (cb - 128)
```

### **Conversion RGB to YCrCb:**

```
y = 0.299r + 0.587g + 0.114b
cb = -0.1687r - 0.3313g + 0.5b + 128
cr = 0.5r - 0.4187g - 0.0813b + 128
```